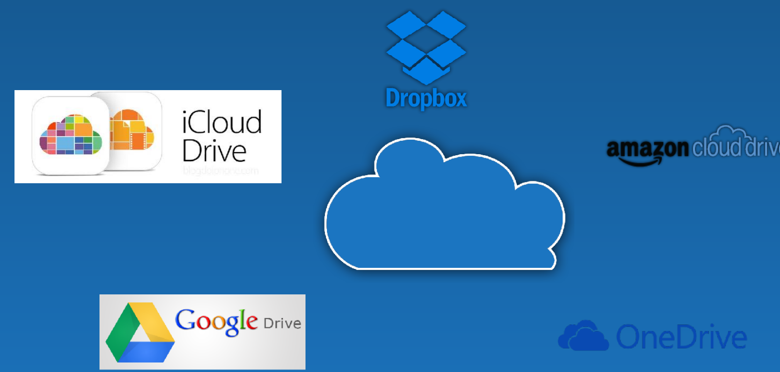
Computations on Encrypted Data and Privacy

David Pointcheval
CNRS - ENS - INRIA

11th International Conference on Provable Security
Xi'an, China - October 23rd, 2017

The Cloud



Anything from Anywhere

One can store
- Documents to share
- Pictures to edit
- Databases to query

and access from everywhere

Security Requirements

As from a local hard drive/server, one expects
- **Storage** guarantees
- **Privacy** guarantees
  - **confidentiality** of the data
  - **anonymity** of the users
  - **obliviousness** of the queries/processing

How to proceed?

Confidentiality vs Sharing & Computations

Classical Encryption allows to protect data

- the provider stores them without knowing them
- nobody can access them either, except the owner/target receiver

How to share the data?

How to compute on the data?

Broadcast Encryption

[Fiat-Naor - Crypto '94]

The sender chooses a target set

Users get **all-or-nothing** about the data

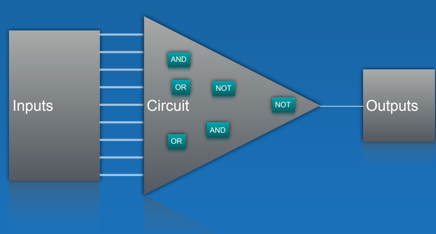
Sharing to a Target Set

but No Computations!

Fully Homomorphic Encryption

[Rivest-Adleman-Dertouzos - FOCS '78]

[Gentry - STOC '09]



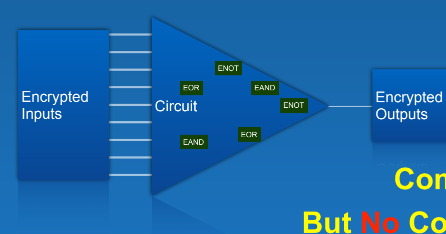
Fully Homomorphic Encryption

[Rivest-Adleman-Dertouzos - FOCS '78]

[Gentry - STOC '09]

FHE allows any computations on encrypted data

But the result is **encrypted** as the inputs!



Computations

But No Controlled Sharing!

Functional Encryption

[Boneh-Sahai-Waters - TCC '11]

The authority generates functional decryption keys DK_f according to functions f

- From $C = \textbf{Encrypt}(x)$, $\textbf{Decrypt}(DK_f, C)$ outputs $f(x)$
- This allows **controlled sharing of data**

Result in clear
for a Specific Function
for Specific Users

Functional Encryption is Powerful

Functional Encryption allows access control:

- with $f_{\mathrm{id}}(x \,||\, y) = (\text{if } y = \text{id, then } x, \text{ else } \perp)$: **identity-based** encryption
- with $f_{\mathrm{G}}(x \,||\, y) = (\text{if } y \in \mathrm{G}, \text{ then } x, \text{ else } \perp)$: **broadcast** encryption

Functional Encryption allows computations:

- any function f: in theory, with *iO* (Indistinguishable Obfuscation)
- concrete functions: inner product

FE: Concrete Case

Student Name	English		CS		Math	
	Written	Spoken	Theory	Practice	Algebra	Analysis
Year 1						
Year 2						
Year 3						

Class	English	CS	Math
Year 1			
Year 2			
Year 3			

Class	English		CS		Math	
	Written	Spoken	Theory	Practice	Algebra	Analysis
Total						

Class	Total
Year 1	
Year 2	
Year 3	

Class	Total
3Years	

- For each student: transcript with all the grades
- Access to partial information for each student
- And even global grades for the class

FE: Inner Product

[Abdalla-Bourse-De Caro-P. - PKC '15 - EPrint 2015/017]

Cells of derived tables are linear combinations $\vec{a_i}$ of the grades $\vec{b}$ from the main table:

$$c_i = \sum_j a_{i,j} b_j = \vec{a_i} \cdot \vec{b}$$

- $\vec{b}$: vector of the private grades, encrypted in the main table
- $\vec{a_i}$: vector of the public coefficients for the cell c_i, defines f_i
- With ElGamal encryption:
  - computations modulo p
  - if grades, coefficients, and classes small enough: DLog computation

FE: Limitations

Initial result: selective security [Abdalla-Bourse-De Caro-P. - PKC '15 - EPrint 2015/017]

But improved to adaptive security [Agrawal-Libert-Stehlé - Crypto '16 - EPrint 2015/608]

Anyway:

- one key limits to one function on any vector 🙂
- a malicious player could ask many functional keys
  - too many keys might reveal the plaintexts… 😞
- a unique sender only can encrypt all the inputs 😞
  - Multi-Input Functional Encryption (MIFE) 🙂

[Goldwasser-Gordon-Goyal-Jain-Katz-Liu-Sahai-Shi-Zhou - Eurocrypt '14 - EPrint 2013/727 - EPrint 2013/774]

IP-FE: Concrete Security?

IP-FE: from $c = \mathbf{E}(\boldsymbol{x})$ and $\mathrm{dk}_{\boldsymbol{y}}$, for n-vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, one gets $\boldsymbol{x}.\boldsymbol{y}$

- n different keys reveal $\boldsymbol{x}$ 😞
- for the indistinguishability between two sets of vectors, the adversary is not allowed to ask keys that trivially tell them appart
  ⇒ if n vectors in the sets, the adversary cannot ask any key! 😞

IP-FE: Too Many Messages/Keys?

IP-FE with Helper: [Dupont-P. - AsiaCCS '17]

- from $c = \mathbf{E}(\boldsymbol{x})$ and $\mathrm{dk}_{\boldsymbol{y}}$, for n-vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, one must ask an helper
- the helper
  - learns as few as possible about the input
    (which ciphertext, which function, which user, etc)
  - limits the number of answers (according to a bound on the inputs)
  - learns nothing about the output
- whereas there are additional interactions
  - no much leakage of information to the helper
  - more reasonable security model 🙂

IP-MIFE: Concrete Security?

IP-MIFE: from $c_1 = \mathbf{E}(\boldsymbol{x}_1), \ldots, c_n = \mathbf{E}(\boldsymbol{x}_n)$ and $\mathrm{dk}_{\boldsymbol{y}}$, one gets $\boldsymbol{x}.\boldsymbol{y}$

- if no ordering: one immediately gets $n!$ linear relations on $\boldsymbol{x}$ 😞
- even with ordering, $c_1 = \mathbf{E}(1, \boldsymbol{x}_1), \ldots, c_n = \mathbf{E}(n, \boldsymbol{x}_n)$
  - if public encryption: only constant-functional keys allowed! 😞
  - if private encryption: mix-and-match attacks 😞

Multi-Client Functional Encryption

- In addition to the ordering, there is a label (or a time period)
  Client $\mathbf{C}_i$ generates $c_i = \mathbf{E}(i, \boldsymbol{\lambda}, \boldsymbol{x}_i)$ for a label $\boldsymbol{\lambda}$
  ⇒ only one ciphertext for each index i and each label $\boldsymbol{\lambda}$

[Goldwasser-Gordon-Goyal-Jain-Katz-Liu-Sahai-Shi-Zhou - Eurocrypt '14 - EPrint 2013/727 - EPrint 2013/774]

- Multi-User Inputs
- Mix-and-match attacks avoided by private encryption
- More reasonable security model 🙂
- But still a unique authority for the functional key generation 😞

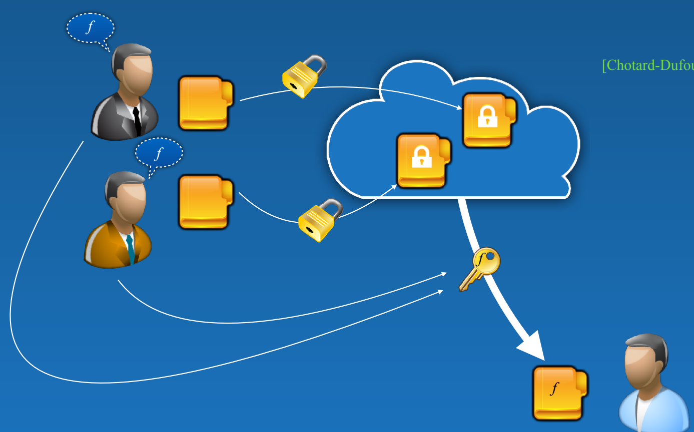
Independent and Untrusted Clients

[Chotard-Dufour Sans-Phan-P. - EPrint 2017/989]

- Senders $(\mathbf{S}_i)_i$ provide sensitive inputs $\boldsymbol{x}_i$ (e.g. financial data)
  in an encrypted way under secret encryption keys ek_i
  $\rightarrow c_i = \mathbf{E}(ek_i, \boldsymbol{\lambda}, \boldsymbol{x}_i)$ for a label $\boldsymbol{\lambda}$ (or every time period)
- For some functions f, an aggregator proposes, as a service,
  to communicate the aggregation $f(\boldsymbol{x})$ for every label $\boldsymbol{\lambda}$,
  thanks to a functional decryption key dk_f
- The senders want to keep control on f
  $\rightarrow dk_f$ is generated by the senders

Decentralized MCFE

[Chotard-Dufour Sans-Phan-P. - EPrint 2017/989]



Decentralized MCFE

[Chotard-Dufour Sans-Phan-P. - EPrint 2017/989]

- **Setup**() $\rightarrow$ secret key sk_i and encryption key ek_i for each sender $\mathbf{S}_i$
  and *mpk*, the master public key
- **Encrypt**$(ek_i, \boldsymbol{\lambda}, \boldsymbol{x}_i) \rightarrow c_i = \mathbf{E}(ek_i, \boldsymbol{\lambda}, \boldsymbol{x}_i)$ for the label $\boldsymbol{\lambda}$
- **DKeyGen**$((sk_i)_i, f) \rightarrow dk_f$
- **Decrypt**$(dk_f, \boldsymbol{\lambda}, \mathbf{C}) \rightarrow f(\boldsymbol{x})$ if $\mathbf{C} = (c_i = \mathbf{E}(ek_i, \boldsymbol{\lambda}, \boldsymbol{x}_i))_i$

- **Encrypt/Decrypt** are non-interactive algorithms
- **Setup/DKeyGen** are interactive protocols between the senders
- **DKeyGen** should be a **one-round** protocol only

ElGamal Encryption

[ElGamal - IEEE TIT '85]

- ElGamal Encryption on $\mathbb{G} = \langle g \rangle$:
  Secret key: $s \in \mathbb{Z}_p$
  Public key: $h = g^s$
  Encryption: $c = (c_0 = g^r, c_1 = h^r \cdot m)$
  Decryption: $m = c_1 / c_0^s$
- Semantically secure under DDH in $\mathbb{G} = \langle g \rangle$
- Multiplicatively homomorphic
- Additive variant: m is replaced by g^m
  but requires discrete logarithm computation
- Encryption of vectors:
  with many h_i and the same randomness

FE: IP with ElGamal

[Abdalla-Bourse-De Caro-P. - PKC '15 - EPrint 2015/017]

Parameters: a group $\mathbb{G} = \langle g \rangle$ of prime order p
Secret key: $\vec{s} = (s_j)_j$, for random scalars in $\mathbb{Z}_p$
Public key: $\vec{h} = (h_j = g^{s_j})_j$
Encryption: $c = g^r$ and $\vec{C} = (C_j = h_j^r \cdot g^{x_j})_j$

$$D = \vec{f} \cdot \vec{C} = \prod_j C_j^{f_j} = g^{r \sum_j f_j s_j} g^{\sum_j f_j x_j} = g^{r \cdot \vec{f} \cdot \vec{s}} g^{\vec{f} \cdot \vec{x}}$$

Functional key: $dk_f = \sum_j f_j s_j = \vec{f} \cdot \vec{s}$
Decryption: $D = c^{dk_f} \cdot g^m \longrightarrow m = \log_g(\vec{f} \cdot \vec{C} / c^{dk_f}) = \vec{f} \cdot \vec{x}$

Because of the common r in the ciphertext,
a unique sender must encrypt the full vector

MCFE: IP with ElGamal

[Chotard-Dufour Sans-Phan-P. - EPrint 2017/989]

Parameters: $\mathbb{G} = \langle g \rangle$ of prime order p, hash function $\mathcal{H}$
Encryption/Secret key: ek$_i$ = sk$_i$ = s_i, for random scalar in $\mathbb{Z}_p$
Encryption: $C_i = \mathcal{H}(\lambda)^{s_i} \cdot g^{x_i}$

$$D = \vec{f} \cdot \vec{C} = \prod_i C_i^{f_i} = \mathcal{H}(\lambda)^{\sum_i f_i s_i} g^{\sum_i f_i x_i} = \mathcal{H}(\lambda)^{\vec{f} \cdot \vec{s}} g^{\vec{f} \cdot \vec{x}}$$

Functional key: $dk_f = \sum_i f_i s_i = \vec{f} \cdot \vec{s}$
Decryption: $D = \mathcal{H}(\lambda)^{dk_f} \cdot g^m \longrightarrow m = \log_g(\vec{f} \cdot \vec{C} / \mathcal{H}(\lambda)^{dk_f}) = \vec{f} \cdot \vec{x}$

Encryption can be performed by independent senders

DMCFE: IP with ElGamal

[Chotard-Dufour Sans-Phan-P. - EPrint 2017/989]

Functional key: $dk_f = \sum_i f_i s_i = \vec{f} \cdot \vec{s} = \vec{1} \cdot \vec{X}$ where $\vec{X} = (X_i = f_i s_i)_i$

- The senders can encrypt $(X_i{=}f_i s_i)_i$
  under another IP-MCFE and the label f
- The aggregator knows the functional key for $(1,\ldots,1)$
- From the ciphertext of $(X_i{=}f_i s_i)_i$, it can extract dk_f

- This would work with a perfect IP-MCFE: any plaintext can be decrypted 🙂
- Here, only small plaintexts can be decrypted: dk_f is large! 😞

DMCFE: IP with Pairings

[Chotard-Dufour Sans-Phan-P. - EPrint 2017/989]

- Two IP-MCFE: $\mathbf{E}_1$ in $\mathbf{G}_1$ and $\mathbf{E}_2$ in $\mathbf{G}_2$
- The senders encrypt the messages x_i with $\mathbf{E}_1$
- The senders encrypt the functional key shares X_i with $\mathbf{E}_2$
- The aggregator knows the functional key for $(1,\ldots,1)$ in $\mathbf{E}_2 \rightarrow$ it gets $g_2{}^{dkf}$
- From $g_2{}^{dkf}$ and ciphertexts of x_i with $\mathbf{E}_1$ in $\mathbf{G}_1 \rightarrow$ one gets $g_T{}^{f \cdot x}$

The discrete logarithm is small: can be extracted!

DMCFE: IP with Pairings

[Chotard-Dufour Sans-Phan-P. - EPrint 2017/989]

Our Decentralised Multi-Client Functional Encryption:

- Selective Security
- Even with Adaptive Corruptions of the Clients/Senders
- Under the classical SXDH assumption
- Efficient **Setup**: generation of the functional key for $(1,\ldots,1)$
- Efficient **DKeyGen** protocol: just one ciphertext sent by each sender

Conclusion

- **Functional Encryption**
  - Ideal functionalities on encrypted data
  - Authority-based functionality
  - Inputs from a unique sender
- **DMCFE**
  - Aggregation of multi-source inputs
  - Functionality under control of the senders